第3章

分周・てい倍・タイミング調整に不可欠なクロック管理機能を使いこなす

――DCMの記述と活用テクニック

松石剛啓

ここでは，FPGAに搭載されているクロック管理機能の使いかたについて解説する．この機能はFPGAファミリによって大きく異なる．今回は，米国Xilinx社のSpartan-3などに搭載されているDCM（Digital Clock Management）を中心に説明する．DCMはクロック・スキュー調整，周波数合成，位相シフトなどに利用できる．（編集部）

Spartan-3には，クロック管理機能としてDCM（Digital Clock Management）がハードウェア・マクロとして組み込まれています．例えば，本誌2005年1月号に付属する基板上のXC3S50には，2個のDCMが組み込まれています．

DCMはFPGAに供給されているクロックから数倍，もしくは数分の1倍の周波数のクロックを生成する場合，あるいはクロックの位相をずらしたいときなどに使用します．例えば，DDR SDRAMコントローラで90°位相のずれたクロックを使用したい場合などには不可欠の機能です．

DCMの特徴

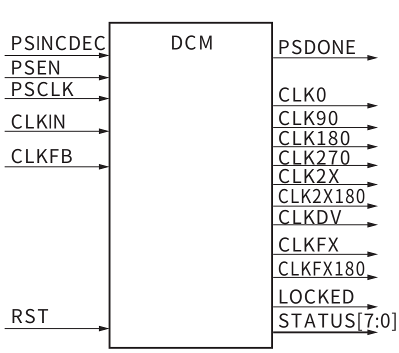
図1にDCMブロックの概要を示します．

DCMには，おもに次の三つの機能があります．

- クロック・スキュー調整
- 周波数合成
- 位相シフト

● クロック・スキュー調整

クロック・スキューとは，クロックの伝搬遅延時間の差などによって発生するタイミングのずれのことです．



(a) DCMブロックの入出力信号

ピン名称	機　能
PSINCDEC	位相調整において，インクリメントまたはデクリメントを選択する
PSEN	可変位相調整を行うため，PSCLKをイネーブルにする
PSCLK	位相シフト調整を同期化させるための信号
CLKIN	オリジナルのクロックを入力する
CLKFB	CLK0またはCLK2Xをフィードバックして入力する
RST	“H”のとき，DCMをリセット
PSDONE	位相調整が完了すると，“H”を出力する
CLK0	CLKINと周波数および位相が一致したクロックを出力する
CLK90	CLKINと同じ周波数で，90°位相がシフトしたクロックを出力する
CLK180	CLKINと同じ周波数で，180°位相がシフトしたクロックを出力する
CLK270	CLKINと同じ周波数で，270°位相がシフトしたクロックを出力する
CLK2X	CLKINと同じ位相で，周波数が2倍のクロックを出力する
CLK2X180	CLKINの2倍の周波数で，180°位相がシフトしたクロックを出力する
CLKDV	CLKINの周波数をCLKDV_DIVIDEというパラメータで分周し，CLKINと位相が一致した分周クロックを出力する
CLKFX	CLKINの周波数をCLKFX_MULTIPLY/CLKFX_DIVIDE倍したクロックを出力する
CLKFX180	CLKFXと同じ周波数で，180°位相がシフトしたクロックを出力する
LOCKED	CLKINとCLKFBの位相がそろうと，“H”を出力する
STATUS	CLKINの状態および位相シフトの状態を出力する

(b) 信号の説明

図1　DCMの概要
Xilinx社のSpartan-3などに内蔵されているDCMブロックの概要を示す．

3

クロック・スキューが大きくなると，セットアップ時間やホールド時間が増加してしまいます．そこで，タイミングを考慮することが重要なアプリケーションでは，スキューをできるだけ小さくする必要があります．

DCMは，そのクロック出力を自分にフィードバックすることで，このスキューを減らすことができます．

● 周波数合成

周波数合成とは，入力クロックに対して，さまざまな周波数のクロックを出力する機能です．

出力周波数の設定は，CLKFX_MULTIPLYというパラメータを分子に，CLKFX_DIVIDEというパラメータを分母にした分数の値で行います．例えば，CLKFX_MULTIPLY

リスト1　DCMエレメントを明示的に記述したVHDLソース・コード

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
-- synopsys translate_off
Library UNISIM;
use UNISIM.Vcomponents.all;
-- synopsys translate_on

entity test_dcm is
    port (
        RST_IN : in std_logic;
        CLKIN_IN : in std_logic;
        LOCKED_OUT : out std_logic;
        CLK0_OUT : out std_logic;
        CLKIN_IBUFG_OUT : out std_logic);
end test_dcm;

architecture RTL of test_dcm is

   signal CLKIN_IBUFG : std_logic;
   signal CLKFB_IN : std_logic;
   signal CLK0_BUF : std_logic;
   signal NET_GND  : std_logic;

   component DCM
    generic(
       CLKDV_DIVIDE : real := 2.0;
       CLKFX_DIVIDE : integer := 1;
       CLKFX_MULTIPLY : integer := 4;
       CLKIN_DIVIDE_BY_2 : boolean := false;
       CLKIN_PERIOD : real := 0.0;
       CLKOUT_PHASE_SHIFT : string := "NONE";
       CLK_FEEDBACK : string := "1X";
       DESKEW_ADJUST : string := "SYSTEM_SYNCHRONOUS";
       DFS_FREQUENCY_MODE : string := "LOW";
       DLL_FREQUENCY_MODE : string := "LOW";
       DSS_MODE : string := "NONE";
       DUTY_CYCLE_CORRECTION : boolean := true;
       FACTORY_JF : bit_vector := X"C080";
       MAXPERCLKIN : time := 1000000 ps;
       MAXPERPSCLK : time := 100000000 ps;
       PHASE_SHIFT : integer := 0;
       SIM_CLKIN_CYCLE_JITTER : time := 300 ps;
       SIM_CLKIN_PERIOD_JITTER : time := 1000 ps;
       STARTUP_WAIT : boolean := false
     );
     port (
       CLKIN : in std_logic;
       CLKFB : in std_logic;
       RST : in std_logic;
       PSEN : in std_logic;
       PSINCDEC : in std_logic;
       PSCLK : in std_logic;
       DSSEN : in std_logic;
       CLK0 : out std_logic;
       CLK90 : out std_logic;
       CLK180 : out std_logic;
       CLK270 : out std_logic;
       CLKDV : out std_logic;
       CLK2X : out std_logic;
       CLK2X180 : out std_logic;
       CLKFX : out std_logic;
       CLKFX180 : out std_logic;
       STATUS : out std_logic_vector (7 downto 0);
       LOCKED : out std_logic;
       PSDONE : out std_logic
       );
   end component;

   component IBUFG
     port (
       I : in std_logic;
       O : out std_logic
       );
   end component;

   component BUFG
     port (
       I : in std_logic;
       O : out std_logic
       );
   end component;

begin

   NET_GND <= '0';

   DCM_INST : DCM
    Generic map (
      CLK_FEEDBACK => "1X",
      CLKDV_DIVIDE => 2.0,
      CLKFX_DIVIDE => 1,
      CLKFX_MULTIPLY => 4,
      CLKIN_DIVIDE_BY_2 => FALSE,
      CLKIN_PERIOD => 20.0,
      CLKOUT_PHASE_SHIFT => "FIXED",
      DESKEW_ADJUST => "SYSTEM_SYNCHRONOUS",
      DFS_FREQUENCY_MODE => "LOW",
      DLL_FREQUENCY_MODE => "LOW",
      DUTY_CYCLE_CORRECTION => TRUE,
      PHASE_SHIFT => 64,
      STARTUP_WAIT => FALSE)
     port map (
      CLKIN => CLKIN_IBUFG,
      CLKFB => CLKFB_IN,
      RST => RST_IN,
      PSEN => NET_GND,
      PSINCDEC => NET_GND,
      PSCLK => NET_GND,
      DSSEN => NET_GND,
      CLK0 => CLK0_BUF,
      LOCKED => LOCKED_OUT);

   CLKIN_IBUFG_INST : IBUFG
     port map (
      I => CLKIN_IN,
      O => CLKIN_IBUFG);

   CLK0_BUFG_INST : BUFG
     port map (
      I => CLK0_BUF,
      O => CLKFB_IN);

   CLK0_OUT <= CLKFB_IN;
   CLKIN_IBUFG_OUT <= CLKIN_IBUFG;

end RTL;
```

DCMを使用するためのライブラリ宣言とパッケージ呼び出し

DCMのコンポーネント宣言

グローバル・入力バッファのコンポーネント宣言

グローバル・クロック・バッファのコンポーネント宣言

DCMのパラメータ設定

位相シフト値設定

BUFG経由でクロックのフィードバック

＝3，CLKFX_DIVIDE＝5のとき，入力周波数×3/5の周波数を生成できます．

● 位相シフト

DCMの出力クロックの位相をシフトできます．1クロック・サイクルの1/256の精度で位相をシフトでき，その量はPHASE_SHIFTというパラメータに－255～＋255の整数を設定することで決定されます．

DCMの使いかた

DCMを使用するには，HDLソース・コード上でDCMのエレメントを明示的に呼び出す必要があります．DCM以外のクロック管理機能を持つデバイスの場合も，それに合わせた記述が必要になります．ターゲット・デバイスに依存しない汎用的な記述はありません．

● エレメント名を直接指定する

CLK_INに入力したクロックの位相を(64/256)×360°シフトさせたクロックを出力するVHDLによる記述の例を**リスト1**に示します．

エレメント名は「DCM」です．分周比などのパラメータを与える際には，genericを使って設定します．分周クロックの生成や位相シフトのような基本機能を使用する際のパラメータを**表1**に示します．**表1**以外のパラメータについては，**リスト1**の記述例で示したとおりに設定しておいてかまいません．詳細については，データシートを参照してください．内蔵数以上のDCMを使用すると，コンパイル(論理合成・配置配線)時にエラーとなるので注意してください．

DCMを使用する際にかならず行わなければならないことがあります．DCMで生成したクロックは，BUFGなどのグローバル・クロック・バッファで駆動してから，CLKFBへ入力したり，ユーザ論理のクロックとして使用しなければなりません(**図2**)．

米国Xilinx社のFPGAには，大きく分けて2種類の配線リソースがあります．一つは通常のローカル配線，もう一つは低キャパシタンスで低スキューのグローバル配線です．通常，Xilinx社のFPGAではクロックにグローバル配線を

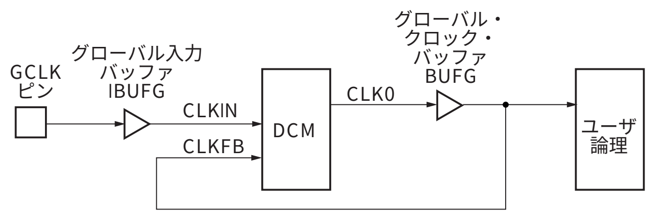


図2　DCMの使いかた
DCMで生成したクロックは，BUFGなどのグローバル・クロック・バッファで駆動してから，CLKFBへ入力したり，ユーザ論理のクロックとして使用しなければならない．

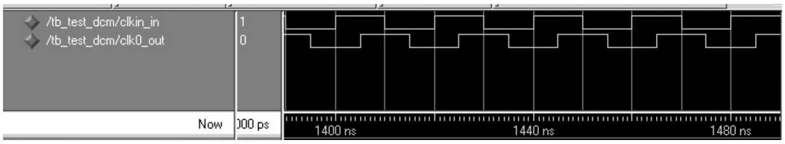


図3
DCMの動作
出力クロックCLK0_OUTは，入力クロックCLKIN_INに対して，64/256（＝1/4）クロック・サイクル分，位相がシフトしている．

表1　DCMのパラメータ

パラメータ名	説　明	値
CLK_FEEDBACK	CLKFB入力へCLK0またはCLK2Xのどちらを入力するかを選択する	NONE，1X，2X
CLKDV_DIVIDE	CLKDV出力の周波数を設定する．CLKDV出力周波数は，CLKIN入力の周波数×(1/CLKDV_DIVIDE)になる	1.5，2，2.5，3，3.5，4，4.5，5，5.5，6，6.5，7，7.5，8，9，10，11，12，13，14，15，16
CLKIN_DIVIDE_BY_2	CLKIN入力の周波数を半分に分周する	TRUE，FALSE
DUTY_CYCLE_CORRECTION	CLK0，CLK90，CLK180，およびCLK270の出力をデューティ50%にする	TRUE，FALSE
CLKFX_DIVIDE	CLKFX出力の周波数を設定する CLKFX出力周波数はCLKFX_MULTIPLY/CLKFX_DIVIDEで設定する	1～32の整数
CLKFX_MULTIPLY	CLKFX出力の周波数を設定する CLKFX出力周波数はCLKFX_MULTIPLY/CLKFX_DIVIDEで設定する	2～32の整数
CLKOUT_PHASE_SHIFT	位相シフトのモードを設定する (シフトなし，固定位相シフト，可変位相シフト)	NON，FIXED，VARIABLE
PHASE_SHIFT	位相シフト値およびシフトする方向を設定する	－255～＋255の整数
CLKIN_PERIOD	CLKIN入力の周期を設定する	単位：ns

使用します．

DCMのシミュレーション結果を**図3**に示します．出力クロックCLK0_OUTは，入力クロックCLKIN_INに対して64/256（＝1/4）クロック・サイクル分，位相がシフトしていることがわかります．

● IPコア生成ツールでカスタマイズ

Xilinx社のFPGA開発ツール「ISE」の有償版では，CORE GeneratorというIPコア生成ツールが使用できます．これを使うと，**図4**のようにDCMのパラメータをGUIで設定できます．また，グローバル・クロック・バッファも自動的に追加してくれます．

Windowsのスタート・メニューから「プログラム」→「Xilinx ISE 6」→「Accessories」→「Architecture Wizard」を選択すると表示されるウィザードを使用しても，DCMモジュールを生成することができます．GUIはCORE Generatorと同じです．

CORE GeneratorでDCMモジュールを作成した後は，そのDCMモジュールを呼び出して使用します．CORE Generatorで作成したDCMモジュールを呼び出すVHDLの記述例を**リスト2**に示します．

リスト1のようにDCMエレメントを明示的に呼び出す際には，genericでDCMのパラメータを書くことになるので，パラメータ名を覚えておかなければなりません．しかし，GUIを使えるCORE Generatorであれば，パラメータ名を覚えなくてもDCMを使用できます．

CLKFX出力を使用して，CLKFX_MULTIPLY＝3，CLKFX_DIVIDE＝5とパラメータを設定し，3/5分周したクロックを出力した例を**図5**に示します．さらに，CLK2X出力を使用して，周波数が2倍のクロックを出力した例を

リスト2　CORE Generatorで生成したモジュールを使うVHDLソース・コードの例

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
-- synopsys translate_off
Library UNISIM;
use UNISIM.Vcomponents.all;
-- synopsys translate_on

entity tst_dcm_top is
    port (
        RST_IN          : in std_logic;
        CLKIN_IN        : in std_logic;
        LOCKED_OUT      : out std_logic;
        CLK0_OUT        : out std_logic;
        CLKIN_IBUFG_OUT : out std_logic);
end tst_dcm_top;

architecture rtl of tst_dcm_top is

component tst_dcm
    port (
        RST_IN          : in std_logic;
        CLKIN_IN        : in std_logic;
        LOCKED_OUT      : out std_logic;
        CLK0_OUT        : out std_logic;
        CLKIN_IBUFG_OUT : out std_logic);
end component;

begin

u_tst_dcm : tst_dcm
    port map (
        RST_IN          => RST_IN,
        CLKIN_IN        => CLKIN_IN,
        LOCKED_OUT      => LOCKED_OUT,
        CLK0_OUT        => CLK0_OUT,
        CLKIN_IBUFG_OUT => CLKIN_IBUFG_OUT);

end rtl;
```

（CORE Generatorで作成したDCMモジュールの呼び出し）

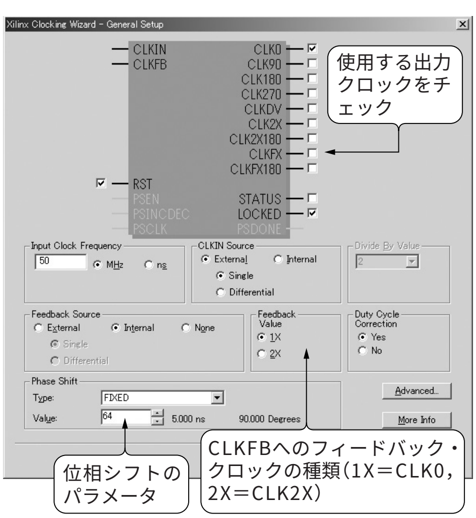


図4　CORE GeneratorによるDCMの設定
DCMのパラメータをGUIで設定できる．

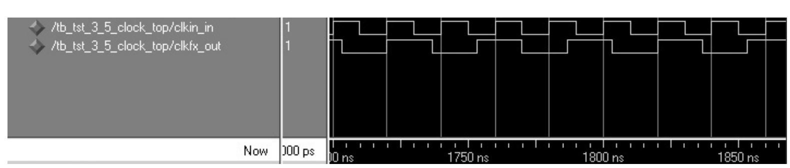


図5　3/5分周のクロック出力波形
CLKFX出力を使用し，CLKFX_MULTIPLY＝3，CLKFX_DIVIDE＝5と設定．

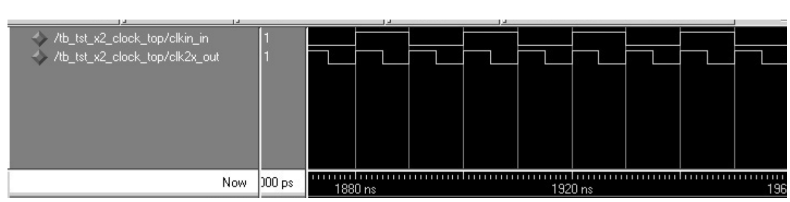


図6　2倍周波数のクロック出力波形
CLK2X出力を使用．

図6に示します．

DCMの活用テクニック

DCMの明示的な呼び出し，あるいはCORE Generatorを使えば，DCMのパラメータを変更できますが，いずれの場合も少しでもパラメータに変更があった場合には論理合成からやり直す必要があります．とくに大規模な設計の場合，位相シフトの設定値をたった＋1するためだけに論理合成を再実行するなどということは，できれば避けたいものです．

● DCMのパラメータを手軽に調整

Xilinx社のISEには，FPGA Editorというツールがあり，これを使うと論理合成を再実行せずにパラメータを変更することができます(**図7**)．FPGA Editorは，Process for Sourceウィンドウの「Implement Design」→「Place & Route」→「View/Edit Routed Design (FPGA Editor)」を選択することによって実行されます．

また，ユーザ制約ファイル(*.ucf)でもDCMのパラメータ設定が可能です．例えば，位相シフトでは，以下の構文を使用します．

```
INST "インスタンス名" PHASE_SHIFT=n
```

(n＝±255の整数)

つまり，DCMのパラメータ設定の方法は，大きく分けて，以下の三つの種類があります．

- FPGA Editorで設定
- ユーザ制約ファイル(*.ucf)で設定
- ソース・コード上の`generic`で設定

さらに，それぞれの設定方法は優先順位が決まっています．FPGA Editorによる設定が高く，ソース・コードの`generic`による設定が低くなります．そこで，位相シフトのように微調整が必要なときには，`generic`で大まかな設定を行っておいて，あとはFPGA Editorでパラメー

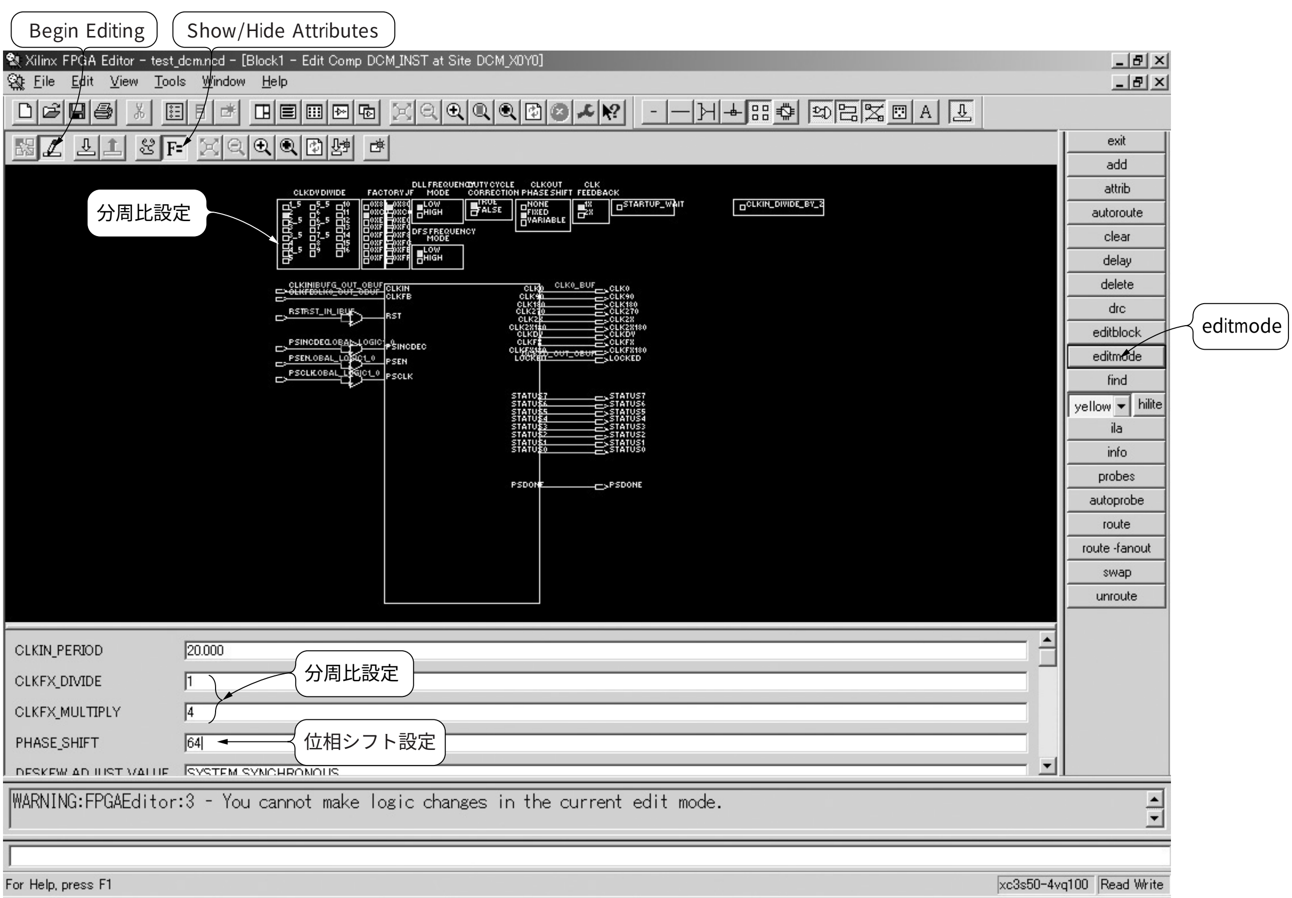
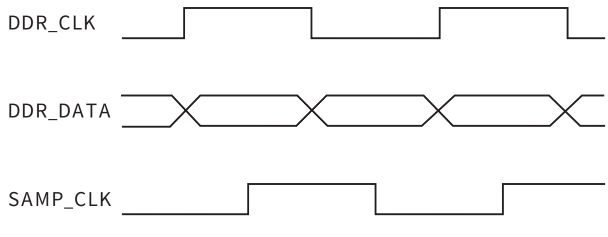


図7　FPGA EditorによるDCMパラメータの変更
FPGA EditorにおけるDCMエレメントの表示．[editmode]ボタンをクリックすると，DCMエレメントのパラメータが変更可能になる．Begin EditingアイコンとShow/Hide Attributesアイコンをクリックして DCMのパラメータを変更する．

図9
DDR SDRAMとDDRコントローラのタイミング
データは正しく読み出されている．

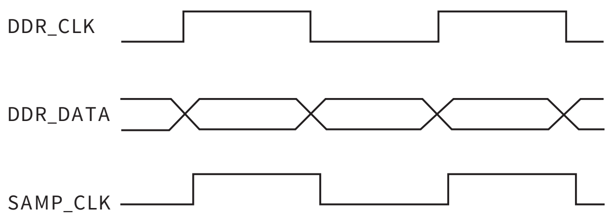


(a) 動作波形



(b) 読み出したデータ

図10
SAMP_CLKをずらす
データを正しく読み出すことができなくなった．



(a) 動作波形



(b) 読み出したデータ

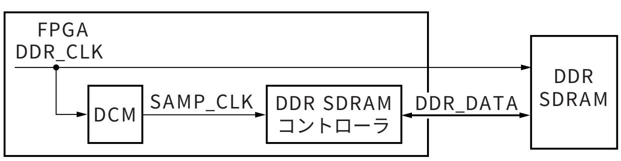


図8　DDR SDRAMとDDRコントローラの接続
FPGA内部は，Xilinx社のDDR SDRAMコントローラとDCMで構成されている．DCMはDDR_CLKと同じ周波数で，位相が90 °ずれたクロック(SAMP_CLK)をDDR SDRAMコントローラへ出力する．DDR SDRAMは，DDR_CLKに同期したデータ(DDR_DATA)を出力する．DDR SDRAMコントローラは，DDR_DATAをSAMP_CLKでサンプリングする．DCMが位相の90 °ずれたクロックを生成することによって，DDR SDRAMコントローラはDDR_DATAが確定しているタイミングでサンプリングできる．

タを変更しながら調整していくとよいでしょう．

● DCMは入出力できる周波数が決まっている

DCMの動作条件を決定するには注意が必要です．

筆者は，FPGAボード上にある16MHzのクロックを使用して，DCMで2倍の周波数のクロックを生成しようとしたことがあります．ところがまったく動きません．使用していたデバイスでは，24MHz以下の周波数を入力できなかったのです．

デバイスおよびDCMの使いかたにより，入出力できる周波数が異なるので，データシートで確認することがたいせつです．

● DCMをSDRAMコントローラに使う

冒頭で，「DDR SDRAMコントローラで90 °位相のずれたクロックを使用する」と説明しました．Xilinx社のEDK(Embedded Development Kit)に付属するDDR SDRAMコントローラは，DDR SDRAMへ入力するクロックに対して位相が90 °ずれたクロックを入力しなければなりません．

そこで，英国Memec社(Memec Insight部門)の「Virtex-II System Board」を使用して実際に動作させてみました．FPGAとDDR SDRAMの接続は**図8**のようになります．DDR SDRAMの制御回路はEDK開発ツールのBase System Builderを使用して設計できます．

正常動作時の波形を**図9**に示します．DCMの位相シフト機能を使用して，90 °の位相ずれを小さくしていくと，**図10**のようにDDR SDRAMコントローラは誤ったデータをサンプリングするようになりました．

参考・引用*文献

(1) Spartan-3 FPGA Family Complete，Data Sheet(ファイル名：ds099.pdf)，Xilinx.
(2) Using Digital Clock Managers(DCMs) in Spartan-3 FPGAs, XAPP422(ファイル名：XAPP462.pdf)，Xilinx.

まついし・たかひろ
(株)ダイヘン

<筆者プロフィール>
現在30歳．ついに三十路になってしまい，かなりショックでした．徳島大学工学部 電気電子工学科卒．6年前にFPGAと出会い，現在もFPGAのディジタル回路設計に夢中．